REMARKS

Claims 9 – 19 , 28, and 30 – 33 are pending. Claims 1-8 have been cancelled without prejudice or disclaimer. New claims 32-34 have been added based on formerly-pending claim 1 and find support in the specification, at least at paragraph [0032]-[0034] and [0041].

Claim 28 has been amended to incorporate limitations from claim 29, which has been cancelled. Claims 19, 30 and 31 have been amended for purposes of clarity.

I. 35 U.S.C. §103 Rejections

In the Office Action mailed September 1, 2009, all currently-pending claims were rejected as allegedly obvious over U.S. Patent Application Publication No. 2004/0230900 (Relyea et al.) in view of U.S. Patent No. 7,559,034 (Paperny et al.).

A proper rejection under 35 U.S.C. §103 requires an evaluation of the scope and content of the prior art, the differences between the claimed invention and the prior art, along with some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness. The present rejections fail to note appreciable differences between the cited references and the subject matter as presently claimed. Additionally, the reasoning articulated in the Office Action does not support the legal conclusion of obviousness. Accordingly, Assignee respectfully submits that the rejections should be withdrawn.

A. Scope and Content of the References

Relyea et al.

Relyea et al. addresses a presumed problem in creating hierarchies of objects programmatically. See, e.g., paragraph [0005]. Relyea et al.'s solution to this problem is defining hierarchies of objects declaratively in an XML-based markup language (see paragraphs

[0007]-[0008]). As noted at paragraph [0009], a parser looks at the markup language to search for classes in which a tag is mapped. Once a class is determined, an attribute name is searched an instance of and object of the class is created. Paragraph [0037] details several examples of CLR objects that can be mapped to tags, including "elements." "Elements" are described as "generally instantiated" during runtime and forming a hierarchy of objects. Next, paragraph [0037] notes that "[t]hese elements remain in the hierarchy. Some elements are instantiated only when needed."

Paragraph [0049] describes how, when the parser encounters a tag, the parser determines which CLR class the tag refers to by searching for a calls within the namespace provided in a definition file for the xmlns definition of the tag. The parser searches in the order that the assemblies and namespace are specified in the definition file, and the parser instantiates an object of the class when it finds a match. Paragraph [0071] describes the flow of parsing the tags in more detail, noting that if a tag is an element tag, the parser looks for a class of the name and instantiates an instance of the corresponding object, noting that the steps 400 to 408 are repeated for each tag until the end of the file.

Paragraph [0064] notes that if an attribute value starts with a "*", the attribute is setting the value equal to a newly instantiated object or an existing object by looking for a name after the *. Paragraph [0064] also states that forward referencing is allowed, and during tree creation if the object referenced doesn't exist yet, the setting of that property is delayed until it does or until the tree is completely created.

Paperny et al.

Paperny et al. is directed to overlaying an object in a window of a software application, using either native functionality of the software application or enhanced functionality (see Abstract). At col. 10, lines 6-49, Paperny et al. notes that a viewer plugin control can be used in a browser to view the overlaid media content. From col. 11, line 28 through col. 12, line 55, Paperny et al. describes instantiation of the viewer plugin in response to events such as document load, mouse click, or during download of content, such as when markup in a web page calls for use of the plugin control (see col. 11, lines 39-44).

B. Differences from the Claimed Subject Matter

The Office Action did not explicitly identify any particular teachings or claimed elements as absent from either reference, although several are present. Several differences are noted below.

For example, neither reference refers to generating or using a descriptor tree having a plurality of nodes. Each independent claim 9, 12, 28 (as amended), and 32 refers to generating a descriptor tree. Claims 12 and 32 further refer to rendering interface elements using descriptor nodes, claim 9 refers to rendering detail objects using the descriptor nodes, and claim 28 (as amended) refers to using the descriptor tree to identify which object(s) are excluded from the subset of objects that are initially instantiated.

Relyea et al. has not been shown to teach generation and use of a descriptor tree based on an internet application and then using the tree as presently claimed. Instead, Relyea et al. relies on parsing an XML file to determine how to create hierarchies of objects. Although Relyea et al. refers to a "tree," it uses the term "tree" and the objects in the hierarchy synonymously (see

paragraph [0007]). The Relyea et al. hierarchy of objects is not used for anything else—the end product appears to be directly created using the markup language (see paragraph [0010]).

The Office Action points to paragraph [0064] of Relyea et al. as teaching the use of hidden descriptor nodes or stacked descriptor nodes describing elements not instantiated/visible at beginning of execution. However, this portion simply refers to the parser supporting use of forward references in the markup language. There is no indication that Relyea et al. teaches use of hidden/stacked nodes in descriptor tree to determine which objects are actually instantiated. Rather, as indicated by the cited portions of Relyea et al., for example the flowchart of Fig. 4, objects corresponding to each element in the XML file are instantiated.

Paperny et al. is not relied on as teaching descriptor trees or any other features noted above as absent from Relyea et al. Paperny et al. is relied on as teaching instantiation of objects in response to user clicking on a hyperlink, banner, or graphical icon. However, Paperny et al. describes instantiation in that context as occurring due to "markup language source code associated with the link," (col. 11, lines 49-55), and not in response to navigation within an application. At the cited portion, Paperny et al. is describing instantiation of a plugin control used to view an object, and not navigation within an application that has already begun execution.

C. The Reasoning in the Office Action does not Support the Conclusion of Obviousness

Considering the claimed subject matter as a whole and differences between the cited references, the claimed subject matter would not have been obvious to one of ordinary skill in the art at the time of invention. Relyea et al. teaches instantiation of objects based on parsing XML. However, Relyea et al. does not teach how one of skill in the art would selectively

instantiate elements. Although paragraph [0037] states that "[s]ome elements are instantiated only when needed," and paragraph [0064] refers to handling a forward reference if "the object referenced doesn't exist yet," Relyea et al. never expands on the concept or explains how "some elements" are instantiated "when needed." Paperny et al. teaches instantiating a plugin application in response to user input, but is not relied on as teaching selective instantiation within the plugin.

Still further, neither reference teaches a descriptor tree having hidden or stacked nodes or the use thereof. Relyea et al. instantiates objects based on parsing markup and makes no mention of the term "descriptor tree," and Paperny et al. is not offered for this purpose. Changing Relyea et al. to use a descriptor tree would represent a major departure from its principle of operation, and would not occur to one of skill in the art based on Paperny et al.

In view of the foregoing, Assignee respectfully asserts that the subject matter of the independent claims would not have been obvious and the claims should be allowable. The dependent claims should be allowable for at least the same reasons.

II. Conclusion

For at least the reasons set forth above, Assignee respectfully requests that the rejections be withdrawn. No fee is believed due with this response. However, if a fee is due, please charge our Deposit Account No. 11-0855.

Respectfully submitted,

Date: 1 December 2009

/Eric G Zaiser/

Eric G. Zaiser
Reg. No. 58,352

KILPATRICK STOCKTON LLP
1100 Peachtree St. NE
Atlanta, GA 30309
404.745.2447 (voice)
404.541.3290 (fax)